Artificial Intelligence 195 (2013) 398–417

Contents lists available at SciVerse ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Exploiting persistent mappings in cross-domain analogical learning of physical domains

Matthew Klenk *, Ken Forbus

Qualitative Reasoning Group, Northwestern University, 2133 Sheridan Road, Evanston, IL 60208, USA

ARTICLE INFO

Article history:
Received 6 July 2011
Received in revised form 9 November 2012
Accepted 13 November 2012
Available online 15 November 2012

Keywords:
Analogical learning
Cross-domain analogy
Cognitive systems
Physics problem-solving

ABSTRACT

Cross-domain analogies are a powerful method for learning new domains. This paper extends the Domain Transfer via Analogy (DTA) method with *persistent mappings*, correspondences between domains that are incrementally built up as a cognitive system gains experience with a new domain. DTA uses analogies between pairs of textbook example problems, or *worked solutions*, to create a domain mapping between a familiar and a new domain. This mapping enables the initialization of a new domain theory. Another analogy is then made between the domain theories themselves, providing additional conjectures about the new domain. After these conjectures are verified, the successful mappings are stored as persistent mappings to constrain future analogies between the domains. We show that DTA plus persistent mappings enables a Companion, the first structure mapping cognitive architecture, to learn the equation schemas and control knowledge necessary to solve problems in three domains (rotational mechanics, electricity, and heat) by analogy with linear mechanics. We provide a detailed analysis categorizing transfer failures. As with people, the most difficult step in cross-domain analogy is identifying an appropriate example. Once an analogous example has been found, DTA successfully transfers the domain knowledge necessary to solve the problem in the new domain 78% of the time. Furthermore, we illustrate how persistent mappings assist in retrieval of analogous examples and overcoming two types of mapping failures.

1. Introduction

Cognitive science research has shown how cross-domain analogies are useful to scientists in producing paradigm shifts [19,25,5] and how students use cross-domain analogies to learn new domains [17]. Textbook authors routinely exploit this aspect of human adaptability when introducing new concepts by providing a useful base domain explicitly and giving hints about correspondences. For example, a typical physics textbook states "the dynamics of rotation is analogous to the dynamics of linear motion" [20]. Students are expected to use this analogy to understand the phenomena and make modeling decisions in rotational mechanics.

Domain Transfer via Analogy (DTA) is a method of learning via cross-domain analogies [28]. DTA consists of four steps: (1) learning the correspondences between the domains, or *domain mapping*, (2) initializing the target domain, (3) extending the target domain, and (4) verifying transferred knowledge. Operating with domain theories, defined in predicate calculus, DTA transfers knowledge from an understood domain to a new domain. In this work, DTA transfers schemas representing equations and control knowledge about their use to new domains. It uses analogies between textbook example problems, *or*

* Corresponding author. Present address: Palo Alto Research Center, 3333 Coyote Hill Rd., Palo Alto, CA, 94304, USA.
E-mail address: Matthew.Klenk@parc.com (M. Klenk).

0004-3702/$ – see front matter
http://dx.doi.org/10.1016/j.artint.2012.11.002

worked solutions, as evidence about what non-identical predicate matches will be productive, re-using those correspondences to transfer equation schemas and control knowledge from the base to the target, creating a new target domain theory.

In previous work [28] we showed that DTA was able to transfer knowledge between kinematic domains. This paper applies DTA to larger and more distant domains. Our previous analyses suggested that learning complex domains by cross-domain analogy does not happen all at once. Instead, the learner's understanding of the analogy grows incrementally. Therefore, we extended DTA with *persistent mappings* [24] to enable the incremental accumulation of knowledge about correspondences between the domains. Learned from successful analogies, persistent mappings constrain subsequent cross-domain analogies, ensuring that a structurally consistent set of knowledge is imported to the target domain. The experiment described here shows that adding persistent mappings to DTA improves performance. By using multiple conditions, we identify several types of failures, and show how persistent mappings address some of them.

We implemented Domain Transfer via Analogy in the Companion Cognitive Architecture [13]. A major hypothesis of the Companion architecture, differentiating it from others, is that analogical processing is central to human reasoning and learning [16]. Given the gap between current AI systems and human abilities, we believe that building human-like systems will likely lead to more flexible AI systems. The Companions architecture includes cognitive models of analogy, similarity-based retrieval and generalization which have been used to account for psychological data. These models have been used in many AI tasks from learning strategy games [24] to everyday physical reasoning [30]. For these reasons, the Companion Cognitive Architecture is an excellent platform for Domain Transfer via Analogy, and in the rest of this article, we refer to the entire system as "the Companion". The Cognitive Systems paradigm [32] investigates intelligence building off of five assumptions: multi-step inference, heuristics, structured knowledge, links to human cognition, and systems-level research. All five appear in this work, placing our efforts squarely within this community. A consequence of doing systems-level research with cognitive models is that Domain Transfer via Analogy could be implemented with different cognitive models of analogical matching and retrieval, providing that the interfaces were the same.

We begin by discussing the DTA algorithm with a running example. In the course of this example, we describe the cognitive models for analogical retrieval and matching used by DTA as well as the representations for physics problems, worked solutions and domain theories. Next, we discuss how persistent mappings enable incremental cross-domain analogy and how they are incorporated into DTA. Then, we describe the experiment and present a detailed analysis of the results. We conclude with a discussion of related and future work.

2. Domain Transfer via Analogy

Domain Transfer via Analogy assumes a known *base* domain and a new *target* domain. In this section, we describe how, given a problem and worked solutions from the base and target domains, DTA transfers knowledge to the target domain. We consider this a *cross-domain analogy* because the base and target domains differ in the object types, quantities, and relations occurring in the problems and domain theories. As a running example in this section, we use linear mechanics as the base domain and electrical systems as the target domain and the problems shown in Fig. 1.[1] We start with describing the representations used by the system.

Linear Mechanics (Base)	Electrical Systems (Target)
Gia-4-1: Estimate the net force needed to accelerate a 1500 kg race car at -5 m/s^2?	Gia-21-37-P: A 75-V emf is induced in a .3 H coil by a current that rises uniformly from 0 to I in 2 ms. What is the value of I?

Fig. 1. Physics problems from two domains that we will use as a running example throughout this paper.

2.1. Physics representation and reasoning

All of the representations in this work are in CycL, the predicate calculus language of the ResearchCyc knowledge base [34]. We use the contents of ResearchCyc, plus our own extensions, to encode the problems, worked solutions, and domain knowledge. Therefore the vast majority of the everyday objects, relations, and events that appear in physics problems were independently developed and are already defined by the 30,000+ concepts and 8000+ predicates in the knowledge base (KB). This reduces the tailorability of our experiments.

2.1.1. Examples of problem and worked solution representations

The problem representations are intended to be direct translations into predicate calculus from natural language problem statements, without any abstraction or reasoning. Consider the linear mechanics problem from Fig. 1. This problem has two *entities*, a race car and an acceleration event. The physical quantities in the problem are denoted using *functions*, e.g., the acceleration of the car during the driving event. To complete the definition, *relations* are used to create *statements*, or *facts*,

[1] The first part of the problem identifier "Gia" indicates where the problem came from [20], the first number indicates the chapter, the second number indicates the problem number, and if there is a "P" on the end it means the problem came from the exercises and not from the worked example section of the chapter.

```
(isa RaceCar41 RaceCar)
(isa Acc41 TransportWithMotorizedLandVehicle)
(objectTranslating Acc41 RaceCar41)
(valueOf (AtFn ((QPQuantityFn Acceleration) RaceCar41) Acc41)
    (MetersPerSecondPerSecond -5))
(valueOf ((QPQuantityFn Mass) RaceCar41)
    (Kilogram 1500))
(isa Gia-Query41 LMPhysicsQuery)
(querySentenceOfQuery
    Gia-Query41
    (valueOf (AtFn ((QPQuantityFn ForceQuantity)
                        RaceCar41 DefOfNetForce)
                   Acc41)
             ?answer))
```

Fig. 2. The representation for problem Gia-4-1 with bookkeeping facts omitted.

```
(isa Gia41-WS-Step2 SubstitutingBindingsForVariables)
(solutionStepUses Gia41-WS-Step2
  (equationFormFor DefOfNetForce
   (mathEquals
       (AtFn ((QPQuantityFn ForceQuantity) ?obj DefOfNetForce) ?evt)
       (TimesFn ((QPQuantityFn Mass) ?obj)
                (AtFn ((QPQuantityFn Acceleration) ?obj) ?evt)))))
(solutionStepUses Gia41-WS-Step2
  (abstractionForObject RaceCar41 PointMass))
(solutionStepUses Gia41-WS-Step2
  (abstractionForObject Acc41 ConstantTranslationAccelerationEvent))
(solutionStepUses Gia41-WS-Step2
...
(solutionStepResult Gia41-WS-Step2
  (equationForSolution Gia41
   (mathEquals
     (AtFn ((QPQuantityFn ForceQuantity) RaceCar41 DefOfNetForce)
           Acc41)
     (TimesFn ((QPQuantityFn Mass) RaceCar41)
              (AtFn ((QPQuantityFn Acceleration) RaceCar41) Acc41)))))
```

Fig. 3. Portion of the representation for step 2 of the worked solution for problem Gia-4-1.

among these terms. This problem is represented as a set of 7 CycL facts, shown in Fig. 2. The first two facts define the two entities in the problem, `RaceCar41` and `Acc41`, and their types. The next fact states that the race car is translating in the driving event. The `valueOf` facts specify the acceleration of the race car during the driving event and its weight. `((QPQuantityFn Acceleration) RaceCar41)` denotes the acceleration of the car, and the function, `AtFn`, is used to refer to the value of a quantity during a period of time, in this case, the acceleration of the car during the driving event, `Acc41`. The last two facts describe the question indicating the sought after parameter, the net force on the race car during the driving event.

The worked solutions are represented at the level of explained examples found in textbooks. They are neither deductive proofs nor problem-solving traces produced by our solver. In instances where the problems did not have worked solutions, they were created to conform to existing worked solutions. Worked solutions consist of a sequence of steps. Each has a *type*, *context*, and *results*. The types of worked solution steps come from an ontology developed in collaboration with Cycorp used in previous work [27]. The context consists of facts from the problem and background knowledge. The results indicate consequences of the worked solution step. Below is an English rendering of the worked solution for the above example.

1. Categorize the problem as a linear mechanics problem.
2. Instantiate the equation for Newton's second law, $F_{\text{net}} = m^*a$, using the race car and the driving event.
3. Solve for the net force on the race car, $F_{\text{net}} = -7500$ N.

A subset of the facts used to encode step 2, which instantiates the definition for net force by substituting entities from the problem into the equation, is shown in Fig. 3. Its context is defined by the statements using the predicate `solutionStepUses`. This context includes an `equationFormFor` statement linking the name of the equation schema being instantiated in the step, `DefOfNetForce`, to the equation, and two `abstractionForObject` statements relating entities in the problem with participant abstraction types in the domain theory. In this case, the entities `RaceCar41` and `Acc41` are modeled as `PointMass` and `ConstantTranslationAccelerationEvent` respectively. Note that CycL is a higher-order logic allowing for statements (e.g., `(abstractionForObject RaceCar41 PointMass)`) to be arguments in other statements. The rest of the context includes facts concerning these entities from the problem case. The results consist of a single statement containing the equation instantiated with the appropriate entities from the problem.

Name	DefOfNetForce
Participants	• ?theObject, type = PointMass • ?theEvent, type = ConstantTranslationAccelerationEvent
Condition	(objectTranslated ?theEvent ?theObject)
Consequences	(equationFormFor DefOfNetForce (mathEquals (AtFn ((QPQuantityFn ForceQuantity) ?theObject DefOfNetForce) ?theEvent) (TimesFn ((QPQuantityFn Mass) ?theObject) (AtFn ((QPQuantityFn Acceleration) ?theObject) ?theEvent)))))

Fig. 4. Definition of Newton's second law equation schema.

```
(frameEquationType ConservationOfLinearMomentum)
(frameEquationType WorkEnergyTheorem-LM)

(preferFrameTypeOver ConservationOfLinearMomentum WorkEnergyTheorem-LM)
```

Fig. 5. Control knowledge indicating frame equations and preferences.

The net force is defined using a function `ForceQuantity`, which takes two arguments, the object on which the force is applied and the source of the force, which is either another entity in the problem or an indicator that this is the net force on the object, as is the case here. The representation for the entire worked solution for this problem consists of 29 facts.

Worked solutions serve as examples from the domain. A central assumption of DTA is that structural similarities between worked solutions from two domains can be used to learn the correspondences between two domains. As we have seen above, the higher-order relationships that provide the connective tissue are domain-independent. They provide the framing which suggests what lower-order content relationships (e.g., `objectTranslating`) might match.

2.1.2. Domain theories for problem-solving

Our physics domain theories consist of equation schemas and problem-solving control knowledge. We represent equation schemas in terms of *participants*, *conditions*, and *consequences*. To instantiate an equation for a given schema, there needs to be a set of entities in the problem that correspond to the types of its participants and satisfy the schema's conditions. For each set of these entities, the equation is instantiated by substituting the entities for each participant in to the consequences. In addition to complete equations, equation schema consequences may define component quantities (e.g., the kinetic energy of one entity is a component of the total kinetic energy in the system). During problem-solving, the Companion instantiates all applicable equation schemas to determine which equations are available.

Fig. 4 shows the definition for the equation schema representing Newton's second law, $F = m^*a$. There are two participants, `?theObject` and `?theEvent`, which must be entities that satisfy the type constraints, the abstractions `PointMass` and `ConstantTranslationAccelerationEvent`, respectively. Furthermore, the conditions of the equation schema must be satisfied in order to instantiate it and conclude its consequences. In this case, it is necessary that `?theObject` be the object translating in `?theEvent`. This knowledge is encoded as a set of predicate calculus facts for the physics problem-solver to use in answering questions.

The physics questions used in this work, like most textbook problems, ask for the values of specific quantities. The problem-solver begins by instantiating equations using the equation schemas of the domain theory. If there is an equation in which the only unknown parameter is the sought quantity, the Companion simply solves the equation to determine the answer. For more complicated problems, the problem-solver makes use of control knowledge. In particular, our problem-solving strategies use *frame equations* to set up problem-solving [40]. Frame equations are central to their domains and serve as starting points for problem-solving. For example, the conservation of linear momentum, $p_{\text{before}} = p_{\text{after}}$, is a frame equation for linear mechanics. The domain specific control knowledge in this work includes knowledge of which equation schemas are frame equations and preferences between them. Fig. 5 shows the representations specifying that the equation schemas for conservation of linear momentum and work energy equations are frame equations and that conservation of linear momentum should be preferred during problem-solving. In addition to the equation schemas, this control knowledge must be transferred as well to solve problems in the target domain.

After selecting a frame equation, the Companion uses strategies of symmetric substitution (e.g., given the work energy equation, $W = Ke_{\text{final}} - Ke_{\text{initial}}$, substitute each of the kinetic energy equations with $1/2mv^2$) and decomposition (e.g., substitute F_{net} with the sum of the component forces). Once the only unknown in the equation is the sought quantity, the Companion invokes algebra routines based on the system described in Forbus and de Kleer [9]. In the case of Gia-4-1, the Companion instantiates the definition of net force equation, identifies that the sought quantity, F_{net}, is the only unknown, and solves the equation to determine that $F_{\text{net}} = -7500$ N.

For concreteness, the entire representations for the worked solution and linear mechanics domain theory are included in Appendix A.

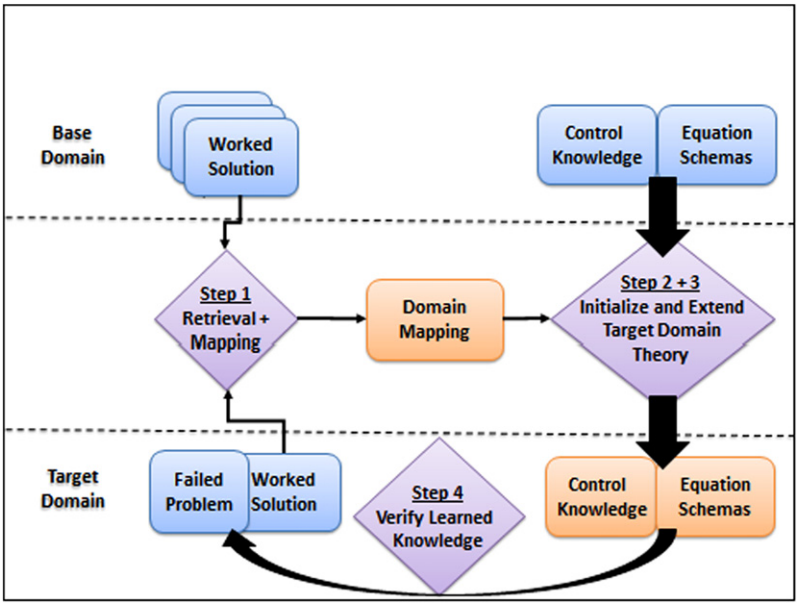

Fig. 6. Domain Transfer via Analogy with blue boxes representing inputs and orange boxes representing learned items. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2.2. Domain Transfer via Analogy: Algorithm

Domain Transfer via Analogy assumes a known base domain theory consisting of equation schemas and control knowledge as well as problem/worked solution pairs. Given a problem that it cannot solve plus its worked solution from a new domain, it learns about the new domain using the following four steps (Fig. 6): (1) learn the domain mapping, (2) initialize the target domain theory, (3) extend the target domain further via analogy, and (4) verify the new knowledge.

With the linear mechanics serving as the base domain, we will illustrate DTA by considering electrical systems as the target domain. Given the problem "A 75-V emf is induced in a .3 H coil by a current that rises uniformly from 0 to I in 2 ms. What is the value of I?" The Companion is unable to generate an answer because its electrical systems domain theory is initially empty. Therefore, DTA is initiated with the worked solution to this problem.

2.2.1. Step 1: Learning domain mapping

Different domains are typically represented with different predicates and conceptual types. Therefore, to perform cross-domain analogy, it is necessary to learn a mapping between the domains. Given a worked solution from the target domain, DTA first finds an analogous worked solution from the base domain and then extracts the correspondences between the worked solutions into a domain mapping. This step relies on two assumptions. (1) The language in which worked solutions are expressed, both in textbooks and our formal versions of them, uses the same terms for the connective tissue of the explanation. That is, predicates like `stepUses` are generic across a wide range of domains. (2) The structure of the explanations mirrors the underlying structure of the domains. In other words, the steps in the worked solution provide partial information about a proof that would derive the answer from what is given plus general domain knowledge.

2.2.1.1. Identify an analogous worked solution from the base domain Using the new target worked solution, we use MAC/FAC [12] to retrieve an analogous worked solution from the base domain. MAC/FAC is a psychological model of similarity-based retrieval that operates in two stages as follows. The inputs to MAC/FAC are a *probe* and a *case library*. In this work, the probe is the target worked solution, and the case library consists of a case for each worked solution from the base domain. In the first stage, Many Are Called (MAC), the probe is compared with every case in the memory pool to select a subset for further consideration. This comparison is done by creating a *content vector* for the probe and each case in the case library. Let Π be the set of all predicates (i.e., relations, functions, and types[2]) in the probe and the case library. A content vector is an n-tuple of numbers, each component corresponding to a particular element of Π. Given a description φ (the probe case or a case in the memory pool), the value of each component in its content vector is proportional to number of times that predicate appears in statements in φ, nominalized so that each content vector is a unit vector. MAC computes a dot product between the probe's content vector and the content vector for each memory case. The highest scoring case and, if they are within 10% of the highest score, up to two more are selected for further consideration in the FAC stage.

[2] While CycL specifies types as *collections*, with the `isa` relation, for MAC/FAC and SME `isa` statements are converted to statements with the collections serving as unary predicates. For example, `(isa RaceCar41 RaceCar)` is converted to `(RaceCar RaceCar41)`.

```
Base Fact:
      (solutionStepUses Gia41-WS-Step2
         (objectTranslating Acc41 RaceCar41))
Target Fact:
      (solutionStepUses Gia2137P-WS-Step2
         (objectActedOn Conduction2137P Coil2137P))
```

Fig. 7. Two facts which result in a match hypothesis between them and match hypotheses for each corresponding argument position

The second stage, Few Are Chosen (FAC), takes as input the descriptions chosen by MAC and computes a full structural match between each of them and the probe using the Structure-Mapping Engine (SME) [6]. SME takes as input two descriptions, a *base* and a *target*. These descriptions are the predicate calculus facts representing the worked solution. The output of SME consists of up to three *mappings*, each representing a structurally consistent interpretation of the comparison between base and target. Each mapping consists of a set of *correspondences* between *items* (i.e., statements, predicates, functions, entities) in the base and target. Mappings also include a *structural evaluation score*, indicating the overall goodness of the match, and *candidate inferences*, which are conjectures about the target using facts from the base which, while unmapped in their entirety, have subcomponents that participate in the mapping's correspondences.

SME creates mappings in four steps: (1) construct local *match hypotheses* describing potential correspondences between base and target items, (2) compute structural evaluation scores for match hypotheses, (3) combine structurally consistent combinations of local matches into global mappings, and (4) compute candidate inferences for each mapping. We use the facts in Fig. 7 as an example to illustrate the algorithm.

The first step operates conceptually in parallel, constructing local match hypotheses by looking for (a) pairs of items that have the same predicate or (b) are suggested by some larger match and are sufficiently alike. The identical relationships prime the pump, i.e. the `solutionStepUses` in Fig. 7. Since corresponding arguments of matched statements have to match (the *parallel connectivity* constraint of structure mapping [15]), SME uses weaker criteria in looking for argument alignments. Entities are always considered to match. In Fig. 7, for example, a match hypothesis is constructed between `Gia41-WS-Step2` and `Gia2137P-WS-Step2`. Non-atomic terms involving non-identical functions can also potentially correspond, since such functions typically represent aspects or dimensions of entities and thus are worth considering. For example, in the domains considered here, the terms involving quantities are such non-atomic terms, where different quantities end up being aligned because the mappings are cross-domain mappings. Moreover, even predicates that are sufficiently similar can be allowed to match as arguments of a larger match. Several criteria have been used in the literature. Here we use *tiered identicality* [15], which allows such matches when there is a close superordinate relation. Returning to Fig. 7, `objectTranslating` and `objectActedOn` are not identical and would not be matched on their own. Since they enable a larger structure to match, and have a close superordinate in the predicate hierarchy (both are subordinates of `preActors` in the ResearchCyc ontology), SME will construct a match hypothesis for these statements, and then for their corresponding arguments in turn.

In the second step, SME computes, conceptually in parallel, a structural evaluation score for each match hypothesis. The purpose of these scores is to provide guidance for the subsequent merge step. Each match hypothesis is initialized with a small initial score. Then *trickle down* is applied, i.e. every match hypothesis contributes score to the match hypotheses that align its arguments. That is, if $W(MH_a)$ is the score associated with a match hypothesis MH_a, MH_b is a match hypothesis that applies to one of MH_a's arguments, and δ is the trickle-down factor, then $W(MH_b)$ is incremented as follows:

$$W(MH_b) \leftarrow \max\{W(MH_b) + \delta W(MH_a); 1.0\}$$

This results in higher scores for correspondences that support larger relational structures. Thus it implements the *systematicity* principle of structure-mapping theory [15], which states that people prefer mappings that are highly interconnected and contain higher-order relations. Returning to Fig. 7, the `Acc41` ↔ `Conduction2137P` match hypothesis has a higher score than the `Gia41-WS-Step2` ↔ `Gia2137P-WS-Step2` match hypothesis.

The third step creates global mappings by finding maximal structurally consistent subsets of match hypotheses. The process starts by identifying those match hypotheses that are not themselves the arguments of any others, and whose substructure is structurally consistent. (In addition to parallel connectivity, structural consistency requires that the correspondences in a mapping be 1:1.) These match hypotheses and all of their argument match hypotheses, recursively, form *kernel mappings*. Kernel mappings have a score, which is simply the sum of the scores of all of the match hypotheses in them. Kernels are ranked and then a greedy algorithm is used to construct global mappings in polynomial time [11,10]. For example, the match hypothesis constructed for the top-level facts in Fig. 7 will provide a kernel mapping, and any other kernel that included a different hypothesis about one of its matches (e.g. `Acc41` and `Coil2137P`) that kernel could not be merged with this one. SME produces up to three mappings, if they are very close (within 10%) of the best, to provide one or two alternatives.

Since greedy algorithms are approximate, it is important to be able to exploit task constraints when possible to improve mappings. SME therefore also allows several kinds of *match constraints* to be specified as part of its input. Two kinds of match constraints are important for this paper. The first are *required correspondences*, i.e. providing a set of correspondences that must hold in any mapping. Such constraints are enforced by using the required correspondences as a starting point for the greedy merge process, thereby eliminating any chance of producing mappings that violate it. The second are *excluded*

Name	DefOfSelfInductance
Participants	• ?theObject, type = Inductor-Idealized • ?theEvent, type = ElectricalConduction-Idealized
Condition	(objectActedOn ?theEvent ?theObject)
Consequences	(equationFormFor DefOfSelfInductance (mathEquals (AtFn ((QPQuantityFn VoltageAcross) ?theObject DefOfSelfInductance) ?theEvent) (TimesFn ((QPQuantityFn Inductance) ?theObject) (AtFn ((QPQuantityFn RateOfCurrentChange) ?theObject) ?theEvent)))))

Fig. 8. Learned equation schema for the definition of self inductance.

correspondences, which indicate items that must not be included in any mapping. These are enforced by eliminating any match hypothesis that contains them.

The final step of the SME algorithm creates *candidate inferences*, which are conjectures about the target justified by the correspondences of the mapping. Candidate inferences are computed by gathering statements in the base that are partially mapped by the correspondences. For each such statement, the substitutions suggested by the correspondences are made. Predicates that are not mapped to something else default to being the same in the inference about the target. Entities that are unmapped are replaced by *analogy skolems*, i.e. a unique new entity postulated in the target because of the mapping. Candidate inferences are the mechanism by which schemas and control knowledge are projected into new domains by DTA.

Returning to MAC/FAC: The FAC stage is simply SME being run, conceptually in parallel, with the structured versions of the descriptions produced by MAC as the base and the probe as the target. Since MAC can only produce up to three outputs, there can be at most three SME runs per retrieval, thereby keeping a tight bound on the computation required. The FAC stage produces at most three cases, by picking the best from the combined output of the SMEs, only selecting multiple cases if their resulting structural evaluation scores are within 10% of the best. In this work, we consider only the case with the best score as the retrieval. Returning to our example, the worked solution for Gia-21-37-P from electrical systems is used as the probe, and MAC/FAC selects the worked solution for Gia-4-1 out of the seven worked solutions of linear mechanics problems.

2.2.1.2. Extract the domain mapping from the analogy between worked solutions Given the analogy between the probe worked solution and the worked solution retrieved by MAC/FAC, we extract the domain mapping as follows. Because SME results in up to three mappings, DTA sorts the mappings by their structural evaluation scores. Beginning with the best mapping, each correspondence is added to the domain mapping when the base entity is mentioned in the base domain theory (i.e., it is a quantity, abstract type, or relation in a condition for an equation schema). This process continues with the rest of the mappings by adding correspondences to the domain mapping that do not violate the one-to-one constraint. For example, if the best mapping had a correspondence between PointMass and Inductor-Idealized and the second mapping had a correspondence between PointMass and ElectricalWire, the domain mapping would only include the mapping between PointMass and Inductor-Idealized. The reason for combining multiple mappings is that the global mapping constraints in SME may preclude the use of otherwise useful local correspondences.

In our example, the worked solution to problem Gia-4-1 is retrieved from memory, and SME creates an analogy between the worked solutions. This analogy results in one mapping. The extracted domain mapping includes abstraction types (e.g., PointMass ↔ Inductor-Idealized), relations (e.g., objectTranslating ↔ objectActedOn) quantities (e.g., ForceQuantity ↔ VoltageAcross), and equation schema names (e.g. DefOfNetForce ↔ DefOfSelfInductance).

2.2.2. Step 2: Initialize target domain theory

When the target domain is empty, DTA uses the domain mapping to initialize the new domain theory. For each equation schema from the base domain mentioned in the domain mapping, DTA attempts to create a corresponding equation schema in the target domain. All of the equation schema's quantities and types must appear in the domain mapping for it to be transferable. If they do, DTA performs the substitutions from the domain mapping on the base equation schema to derive a new equation schema. The new equation schemas are added to the target domain theory.

In our example, there is only one equation schema in the domain mapping, DefOfNetForce. The linear mechanics domain theory includes facts defining the equation schema for DefOfNetForce (shown in Fig. 4) and that it is frame equation. Each of these facts is transferred to the electrical domain theory by substituting each entity that appears in the domain mapping with its corresponding entity from the target domain. This results in an equation schema for the definition of self inductance $V = L^* \, \mathrm{d}i/\mathrm{d}t$, shown in Fig. 8, as well as the control knowledge indicating it is a frame equation.

2.2.3. Step 3: Extend target domain theory

DTA extends the target domain theory through a second cross-domain analogy using the base and target domain theories themselves. The domain theories consist of the facts representing the equation schemas and the control knowledge

- Given: Target Problem, *tp*, Target Worked Solution, *tws*, Case Library of Base Worked Solutions, $CL = bws_1, \ldots, bws_i$, base domain theory, *baseDT*, a target domain theory, *targetDT* (possibly empty), and a set of persistent mappings, *pm* (possibly empty)

1. Learn the domain mapping, *dm*
   - Retrieve analogous base worked solution bws_j using MAC/FAC with *tws* as probe and *pm* as required correspondence constraints
   - Use SME to create a set of analogical mapping, $am_{1..k\ k<4}$, mapping with bws_j as the base and *tws* as the target constrained by *pm*
   - For each *am* sorted by structural evaluation score
     - For each correspondence in *am* if the base entity is part of *baseDT* and not in *dm*, then add the correspondence to *dm*
2. If *targetDT* == null, Initialize *targetDT*
   - For each base equation schema in *dm*, create a target equation schema and add it to *targetDT* by replacing each entity with the corresponding entity in *dm*
3. Extend *targetDT*
   - Create an analogy between *baseDT* and *targetDT* constrained by *dm*
   - For each equation schema in the candidate inferences, add it to *targetDT*
4. Verify *targetDT* by attempting to solve *tp*
   - If unsuccessful, then, erase newly added facts from *targetDT* and *dm*, else, $pm = dm$

Fig. 9. DTA extended with persistent mappings.

necessary for problem-solving. Recall that SME optionally takes match constraints to guide its mapping process. The correspondences extracted from the first mapping are used to generate required correspondence constraints, thus ensuring that they are respected in attempting to project new information to the target. Also, excluded correspondence constraints are created for each equation schema in the target domain that does not participate in the domain mapping, to prevent them from interfering with subsequent matches.

This second cross-domain analogy produces candidate inferences concerning additional equation schema and control knowledge that can be imported into the target domain theory. For each equation schema mentioned in the candidate inferences, if all of its participant types and quantities participate in the analogy, DTA adds the new equation schema defined by the candidate inferences to the target domain theory. Candidate inferences concerning control knowledge that refer to mapped equation schemas are also imported to the target domain.

In our example, DTA creates an analogy between the linear mechanical and electrical domain theories themselves. None of the candidate inferences from this analogy include additional equation schemas or control knowledge; therefore no knowledge is transferred in this step.[3]

2.2.4. Step 4: Verify learned knowledge

While powerful, cross-domain analogies are risky and frequently contain invalid inferences. Consequently, DTA verifies the newly proposed knowledge by re-attempting the target problem with the newly extended target domain theory. If this problem is solved correctly (i.e., its answer is the same as the answer in the provided worked solution), DTA assumes that the learned knowledge is correct. Otherwise, DTA forgets the learned domain mapping, equation schemas, and control knowledge. Attempting to find other analogs is of course possible, but currently our system stops after the first failure. In future work, we plan on placing this choice under the Companion's control to decide if additional attempts would be helpful.

After a successful transfer, the equation schemas and control knowledge are available for reasoning about future problems. In our example, the Companion verifies this knowledge by successfully solving Gia-21-37-P. Therefore the Companion assumes the new electrical domain theory is correct.

In the next section, we describe how we extended DTA with persistent mappings to enable a Companion to incrementally learn a new domain through cross-domain analogy.

3. Extending DTA with persistent mappings

Persistent mappings allow complex cross-domain analogies to be incrementally constructed as additional knowledge about the target domain becomes available. As shown in Fig. 9, persistent mappings are an additional input to the DTA algorithm and are updated with each successful transfer. When presented with a worked solution from a new domain (as in Section 2), the persistent mappings are empty. After a successful transfer, the domain mapping is stored as persistent mappings to constrain future cross-domain analogies. As the Companion is exposed to new worked solutions for failed problems in the target domain, the persistent mappings constrain the cross-domain analogy process as follows. In Step 1 of DTA, the persistent mappings are used as required correspondence constraints in MAC/FAC and SME to constrain possible mappings between worked solutions. Recall that the MAC part of MAC/FAC retrieval performs a dot product between the probe content vector and the content vector for each case in the case library. Required correspondences alter this process by overwriting the probe's content vector as follows. For each set of predicates that appear in the persistent mappings, replace the value of the target predicate with the value of the base predicate. Then during the FAC stage, the persistent mappings

[3] While there was no knowledge transferred in this example, control knowledge indicating preferences between learned equation schemas was transferred in this experiment. In previous work, we demonstrated that additional equation schemas could be transferred, enabling the transfer of knowledge to go beyond items which were referenced in the worked solution [28].

Linear Mechanics (Base)	Electrical Systems (Target)
Phy-1: When a 13.2-kg mass is placed on top of a vertical spring, the spring compresses 5.93 cm. Find the force constant of the spring.	Gia-17-24-P: How much charge flows from a 12-V battery when it is connected to a 7.5 micro-F capacitor?

Fig. 10. An example of incremental cross-domain analogical learning between linear mechanical and electrical systems.

are simply used as required correspondence constraints in SME. Thus persistent mappings should improve the retrieval of analogous worked solutions as well as the utility of the resulting domain mapping.

After the successful iteration of DTA above, the Companion is presented with a new problem from electrical systems, Gia-17-24-P (shown in Fig. 10). While the Companion has a nascent electrical domain theory, it is insufficient for this problem. Consequently, the Companion fails to arrive at the correct solution and invokes DTA with the worked solution to this problem and linear mechanics as the base domain theory.

Unlike the previous iteration, the Companion has a target domain theory consisting of a single equation schema and a set of persistent mappings between these domains (e.g., `PointMass` ↔ `Inductor-Idealized`, `objectTranslating` ↔ `objectActedOn`, `ForceQuantity` ↔ `VoltageAcross`). This guides retrieval and mapping, which selects the worked solution to problem Phy-1 and constructs a mapping that builds on its knowledge about the domains. The resulting domain mapping includes the persistent mappings and the correspondences between the worked solutions to Phy-1 and Gia-17-24-P (e.g., `Distance` ↔ `Charge` and `Compliance-Linear` ↔ `Capacitance`). Because there is already an electrical domain theory, it is not necessary to initialize it. In the third step, the domain mapping constrains the analogy between the linear mechanics and the electrical systems domain theories. DTA uses the resulting candidate inferences from the analogy between the domain theories to extend the target domain theory with the equation schema for the definition of charge on a capacitor, $V = q/C$. In the final step, the Companion verifies this knowledge by successfully solving Gia-17-24-P. Therefore, the Companion extends the persistent mappings between linear mechanics and electrical systems with the new correspondences.

As the Companion incrementally extends the target domain theory using DTA, the persistent mappings maintain a one-to-one correspondence between the base and target domains. That is, each item in the base corresponds with only one item in the target and vice versa. When there are no persistent mappings, then a successful iteration of DTA will result in a set of one-to-one persistent mappings. When there are persistent mappings, then the mappings produced in steps 1 and 3 of DTA will conform to the mappings. Therefore, the correspondences added to the persistent mappings by a successful iteration of DTA cannot violate the one-to-one constraint between the domains. While constraining the analogies in this way enables DTA to find useful correspondences between dissimilar worked solutions, if there is not a one-to-one correspondence between the abstractions, relations and quantities that make up the base and target domains, then DTA will be unable to learn the entire target domain. We discuss the implications of this in Section 4.4.3.

Having defined persistent mappings and shown how they enable incremental cross-domain analogy as additional examples become available, we next turn to an experiment that provides evidence that this technique is useful, and analyze its strengths and weaknesses.

4. Dynamical analogies

In previous work, we demonstrated that a Companion using DTA without persistent mappings could rapidly learn rotational kinematics by analogy with linear kinematics and vice versa [28]. While an important result, that experiment had four limitations:

- Linear and rotational kinematics are highly similar domains.
- Both domains were about kinematics.
- Only equation schemas were transferred.
- The domain theories did not include any non-analogous elements.

To address these limitations, we created a new corpus of domains based on Olson's *Dynamical Analogies* [39] including linear mechanical, rotational mechanical, electrical, and thermal systems. The dynamical analogy domains differ from the kinematics domains used in the previous experiment in several important dimensions. First, the new domains include superficially dissimilar domains such as mechanical and electrical systems. Second, the dynamical analogy domains include more phenomena than the kinematics scenarios. This makes them more challenging, since a single cross-domain analogy between two worked solutions does not include all of the entities from the base and target domain theories. Third, the complexity of the problems was increased, requiring control knowledge to be included in the domain theories. Fourth, these domains include non-analogous elements (e.g., equations to calculate the moment of inertia of a point in rotational mechanics, and nothing corresponds to kinetic energy in thermal systems). Table 1 shows the analogous quantities from the domains.

This experiment evaluates DTA's performance on the dynamical analogy domains to address the following questions:

- Can DTA transfer the analogous knowledge necessary to solve problems in this broader set of domains?

Table 1
Dynamical analogy domains aligned by analogous quantities.

Linear	Rotational	Electrical	Thermal
Force $[F]$	Torque $[T]$	Voltage across $[V]$	Temperature difference $[T]$
Speed $[v]$	Rate of rotation $[\omega]$	Electrical current level $[i]$	Heat flow rate $[q]$
Linear deflection $[x]$	Rotational deflection $[\beta]$	Electrical charge $[q]$	Thermal energy $[H]$
Mass $[F = ma]$	Moment of inertia $[T = J\alpha]$	Inductance $[V = L\,\mathrm{d}i/\mathrm{d}t]$	n/a
Linear momentum $[p = mv]$	Rotational momentum $[p = J\omega]$	n/a	n/a
Linear kinetic energy $[Ke = 1/2mv^2]$	Rotational kinetic energy $[Ke = 1/2J\omega^2]$	Inductance energy $[\text{Energy} = 1/2Li^2]$	n/a
Linear compliance $[F = x/C]$	Rotational compliance $[T = \beta/C]$	Electrical capacitance $[V = q/C]$	Thermal capacitance $[T = H/C]$
Translational elastic potential $[\text{EPE} = .5(x^2)/C]$	Rotational elastic potential $[\text{EPE} = .5(\beta^2)/C]$	Capacitance energy $[\text{Energy} = .5(q^2)/C]$	n/a
Linear damping $[F = bv]$	Rotational damping $[T = D\omega]$	Electrical resistance $[V = Ri]$	Thermal resistance $[T = Rq]$
Power $[P = Fv]$	Power $[P = T\omega]$	Power $[P = Vi]$	n/a

- When retrieval fails, does providing the Companion with the analogous base worked solution lead to successful transfer?
- What are the effects of persistent mappings in learning domain mappings and aiding retrieval?

4.1. Materials

The problems were selected from five physics resources [45,20,7,38,26] with the following goals. First, each of the quantities in Table 1 was included in the problem set. Second, the problems were limited in complexity to require at most three different physics equations. This is consistent with the standard practice of using more basic problems when introducing learners to new subjects. Third, problems were favored which provided a worked solution. For problems that did not have corresponding worked solutions, we created them. Fourth, problems involving calculus were simplified to enable our algebra system to solve them without modification. Table 2 contains natural language representations of the problems aligned by their underlying analogous principles.

Predicate calculus representations for the 22 problems and worked solutions were created: 7 from linear mechanics, 7 from rotational mechanics, 6 from electrical systems, and 2 from thermal systems.[4]

4.2. Method

The base domain for this experiment is linear mechanics, because it is the dynamical analogy domain most students learn first. The seven linear mechanics problems and worked solutions were added to the case library. The base domain theory consists of the eleven equation schemas and control knowledge necessary to solve these seven problems. The target domains for this experiment are rotational, electrical, and thermal systems. It would be impossible to learn these domains entirely by analogy. Therefore, we provided the companion initial target domain theories including the non-analogous equation schemas necessary to solve the problems (e.g., equations for computing moment of inertia). This includes 4 rotational, 0 electrical, and 2 thermal equation schemas. None of the target problems are solvable with only these equation schemas.

This experiment evaluates DTA on a per problem basis in four independent conditions. In the first condition, the entire DTA algorithm was run on each problem. To isolate the effects of the retrieval mechanism, the Companion was only allowed one attempt to retrieve a worked solution from the base domain. In the second condition, each problem was presented with the correct retrieval from linear mechanics. The third and fourth conditions address the effects of persistent mappings. In the third condition, each problem was presented along with the persistent mappings resulting from a successful transfer using DTA on the most similar problem. For example, for problem Gia-21-37-P, the persistent mappings were provided from a successful transfer between Gia-6-3 and Gia-21-46-P. In the fourth condition, the persistent mappings and the correct retrieval were provided.

We measured two aspects of performance, the Companion's ability to solve the problem in the target domain after DTA and if retrieval selected the analogous base worked solution. For each problem, the problem was scored correct if the Companion was able to solve the problem after transfer and the problem's retrieval was scored correct using the rows in Table 2. Therefore, each problem has only one correct retrieval out of 7 worked solutions in the linear mechanics case library.

The goal of this experiment was to assess DTA's ability to learn rotational mechanical, electrical, and thermal systems through cross-domain analogy with linear mechanics. Using the four conditions, the retrieval, mapping and persistent mapping components were evaluated independently.

[4] The representations for all of the problems and worked solutions can be found at http://www.matthewklenk.com/dissertation.htm.

Table 2
Dynamical analogy problems aligned by central equations.[a]

Linear	Rotational	Electrical	Thermal
Gia-4-1: Estimate the net force needed to accelerate a 1500 kg race car at -5 m/s^2?	Gia-8-9: A 15 N force is applied to a cord wrapped around a 4 kb wheel with a radius of 33 cm. The wheel is observed to accelerate uniformly from rest to reach an angular speed of 30 rad/s in 3 s, if there is a frictional torque of 1.1 Nm, determine the moment of inertia of the wheel.	Gia-21-37-P: A 75-V emf is induced in a .3 H coil by a current that rises uniformly from 0 to I in 2 ms. What is the value of I?	n/a
Gia-7-2: A 10,000 kg railroad car traveling at a speed of 24 m/s strikes an identical car at rest. If the cars lock together as a result of the collision, what is their common speed afterwards?	Gia-8-12: A mass attached to the end of a string revolves in a circle on a frictionless tabletop. The other end of the string passes through a hole in the table. Initially, the ball rotates with a speed of 2.4 m/s in a circle of radius .8 m. The string is then pulled through the hole so that the radius is reduced to .48 m. What is the speed of the mass now?	n/a	n/a
Gia-6-3: A 145 g baseball is thrown with a speed of 25 m/s. What is the work done if it is starting from rest?	Gia-8-40-P: A centrifuge rotor has a moment of inertia of 4e10^{-2} kg m^2. How much energy is required to bring it from rest to 10,000 rpm?	Gia-21-46-P: How much energy is stored in 40 mH inductor at an instant when the current is 12 A?	n/a
Phy-1: When a 13.2-kg mass is placed on top of a vertical spring, the spring compresses 5.93 cm. Find the force constant of the spring.	She-2-16: A simple torsion bar is formed by a cylindrical steel rod has a diameter of .25 inch and a length of 3 inches and is fixed to a rigid support at one end. The end rotations 1 radian when a 20 N force is applied, Compute the inertia of the rod?	Gia-17-24-P: How much charge flows from a 12-V battery when it is connected to a 7.5 micro-F capacitor?	Gia-14-1: How much heat is required to raise the temperature of an empty 20-kg vat made of iron from 10C to 90C?
Gia-6-28-P: A spring has a spring constant of 380 N/m. How much must this spring be compressed to store 60 J?	She-2-18: An ideal torsion spring is attached to the shaft of a gear. A torque, T, is applied to the gear twisting the spring. Assuming that $K = 11$ N m/rad. Compute the energy stored in the spring when it is displaced by 2 rads.	Gia-17-7: A 12-V battery is connected to a 20 micro-F capacitor with uses a paper dielectric, how much electric energy can be stored in the capacitor?	n/a
Oga-3-8: Given a speed of 20 m/s what is the force applied by a series of dampers with damping constants of 7 N s/m and 5 N s/m?	Oga-6-12: A Cylinder is rotating with a speed of 15 rad/s what is the force applied from a dashpot (b = 2 N s/rad)?	Gia-18-2: A plate on the bottom of a small tape recorder specifies that it should be connected to that it should be connected to a 6 V and will draw 300 mA, what is the resistance of the recorder?	Gia-14-8: A major source of heat loss from a house is through the windows. Calculate the rate of heat flow through a glass window 2 m $\times$ 1.5 m and 3.2 mm thick if the temperatures at the inner and out surfaces are 15C and 14C.
Gia-6-47-P: If a car generates 15 hp when traveling at a steady 100 km/h, what must be the average force exerted on the car due to friction and air resistance?	Rea-286: The drive shaft of an automobile rotates at 377 rad/s and transmits 59.6 W from the engine to the rear wheels. Compute the torque developed by the engine.	Gia-18-6: An electric heater draws 15 A on a 120 V line. How much power does it use and how much does it costs per month 30 days if it is operated 3 hrs/day and the electric company charges $.08 per Kwh?	n/a

[a] Problem sources are indicated by their prefixes. The first number indicates the chapter and the second problem indicates the example within that chapter. The 'P' designation is used for problems from the end of the chapter. "Gia" are from [20], "Oga" are from [38], "She" are from [45], "Rea" are from [7] and "Phy" are from [26].

4.3. Results

Table 3 shows the results of the four trials for each of the three transfer domains.

In condition 1, DTA transferred the necessary equation schemas and control knowledge to solve 3 out of 15 problems from the target domains (20%). While the Companion solved problems from rotational mechanical and electrical domains,

Table 3
DTA results using linear mechanics as the base domain.

Condition	Rotational mechanical (7)		Electrical systems (6)		Thermal systems (2)		Total (15)	
	Cor. (%)	Ret. (%)	Cor. (%)	Ret. (%)	Cor. (%)	Ret. (%)	Cor. (%)	Ret. (%)
1. Original DTA	2 (29%)	3 (43%)	1 (17%)	2 (33%)	0 (0%)	0 (0%)	3 (20%)	5 (33%)
2. Given base worked solution	6 (86%)	n/a	4 (67%)	n/a	2 (100%)	n/a	12 (80%)	n/a
3. Given persistent mapping	3 (43%)	3 (43%)	1 (17%)	3 (50%)	0 (0%)	0 (0%)	4 (27%)	6 (40%)
4. Given persistent mapping and base worked solution	7 (100%)	n/a	4 (67%)	n/a	2 (100%)	n/a	13 (87%)	n/a

it failed to solve either of the thermal system problems. This result is not as bad as it first appears. Recall that we only allowed the Companion one retrieval attempt. When the analogous worked solution was retrieved, the Companion was able to solve the target problem 60% of the time. In condition 2, where DTA was provided the correct retrieval from memory, the Companion successfully solved problems from all three of the domains. DTA transferred useful equation schemas and control knowledge for 12 out of the 15 problems (80%). In the third condition, the Companion was provided with the persistent mappings resulting from a successful transfer of the closest target problem. After retrieving 6 out of 15 (40%) correctly, DTA successfully transferred knowledge to solve 4 (27%) of the problems. Unlike condition 1, the retrieval result for condition 3 is statistically significant from random retrieval ($p < .05$). In the final condition, DTA was provided with both the persistent mappings and correct retrieval. In this case, the Companion correctly solved 13 of the 15 (87%) problems.

4.4. Discussion

First, these results indicate DTA is able to transfer domain knowledge from linear mechanical systems to solve problems from related domains. In addition to successful transfer to rotational mechanics, DTA was able to transfer linear mechanical knowledge to the superficially dissimilar domains of electrical and thermal systems when the analogous worked solution was retrieved. In this experiment, DTA transfers not only equation schemas but also control knowledge. While the analogous portion of these domains contains 7 equations relating 11 quantities, there are 33 total equations and 56 total quantities used in the 4 domain theories and 22 worked solutions. In the next sections, we analyze the results with respect to the retrieval, transfer and persistent mapping aspects of the algorithm in detail.

4.4.1. Retrieval

The primary cause for DTA's failures was the inability to retrieve an analogous worked solution. This is consistent with psychological findings regarding the difficulties in the spontaneous retrieval of cross-domain analogies [21,18]. Given a known base domain, DTA retrieved the analogous worked solution only 33% of the time. By providing persistent mappings, this performance improved to 40%, which is statistically significant compared to chance ($p < .05$).

While MAC/FAC includes both superficial and structural similarities in each stage, the structural similarity is not immediately apparent because the relations defining the worked solution structure carry the majority of the weight in the content vector. Consequently, every retrieval failure occurred during the MAC stage of MAC/FAC. Recall that the MAC stage computes a dot product between content vectors for each of the cases in the case library and the probe case. The content vectors are automatically computed and each component has a strength proportional to the number of occurrences of individual predicates. Looking at the content vectors more closely, the majority of the highest weighted entries are relations concerning the definition of worked solution steps. Of the rest of the entries, few are shared across the domains. Rotational mechanics shares some motion predicates with the base domain, contributing to improved retrieval. Also, persistent mappings improved retrieval slightly by altering the probe's feature vector to include additional predicates from the base domain using the required correspondence constraints. Every time the analogous worked solution was selected in the MAC stage, it was returned as the retrieval. SME's preference for deep structural matches and its ability to move beyond identical relations when suggested by larger structure enables FAC to select the appropriate analog. While we could run SME against every case in memory, this clearly would not scale to larger case libraries. Therefore, an important aspect of future work is to put the decision to perform additional retrievals under the Companion's control.

Combining the results from conditions 1 and 3, the 37% success rate of retrieval was statistically better than chance ($p < .05$). For this experiment, the Companion was only allowed one retrieval attempt. One method to improve performance would be to allow multiple attempts. The verification of transferred knowledge identifies non-analogous retrievals and prevents failed transfers from corrupting the target domain theory.

4.4.2. Mapping and transfer

Given the cross-domain nature of these analogies, the analogical mappings between worked solutions and domain theories were very successful. Our assumption that the structure of explanations mirrors the underlying structure of the domains holds for these problems. Of the 41 problems in which DTA used the correct retrieval, transfer was successful on 32 (78%)

of them. Through analyzing the failures, we can better understand the limits of DTA. The 9 transfer failures can be divided into three types: *merge failures*, *one-to-one constraint failures*, and *incomplete mapping failures*.

Merge failures occur when the mapping fails to include a necessary correspondence because a competing correspondence is already in the mapping. During SME, local match hypothesis are merged into structurally consistent global mappings. For example, the worked solution of Gia-6-3 includes two relations connecting the ball to the throwing event: `objectThrown` and `objectTranslating`. The analogous rotational mechanics problem, Gia-8-40-P, has one relationship linking the rotor to its rotation event: `objectRotating`. Due to the one-to-one constraint, `objectRotating` can only map to `objectThrown` or `objectTranslating`. If it maps to `objectThrown`, the subsequent cross-domain analogy will fail because `objectTranslating` is in the condition in the analogous equation schema.

The second cause of mapping failures are one-to-one constraint violations in the instantiations of analogous equations. For example, the worked solution to problem Gia-6-3 uses the equation schema for linear kinetic energy ($Ke = \frac{1}{2}mv^2$) which includes three participants: the object, the movement event, and the time point of the measurement. The three participants are `Ball-6-3`, `Throwing-6-3` and `(EndFn Throwing-6-3)` respectively. The analogous electrical problem, Gia-21-46-P, uses the equation schema for inductance energy ($E = \frac{1}{2}Li^2$), which also has three participants: the inductor, the induction event, and the time point of the measurement. In this case, the induction event is the same entity, `Induction-21-46-P`, as the time point. The one-to-one constraint requires only 2 of the base entities to map to the 2 target entities. Consequently, only two of the three participant types for the linear kinetic energy equation schema appear in the mapping. Recall that if all of an equation schema's participant types are not in the domain mapping, then the equation schema cannot be transferred via DTA.

The third cause of mapping failures is an incomplete mapping. This occurs when the mapping with the analogous worked solutions transfers only some of the equation schemas required to solve the target problem. This error occurred during the analogy between electrical problem Gia-17-7 and the linear mechanics problem Gia-6-28-P. Gia-17-7 requires equation schemas for both the definition of electrical capacitance, $V = q/C$, and the definition of capacitor energy, $Ce = \frac{1}{2}(q^2)/C$. Gia-6-28-P only contains the analogous equation for capacitor energy, $\text{EPE} = \frac{1}{2}(x^2)/C$. Therefore, while it successfully transfers this equation schema, the verification step fails because the Companion does not have an equation schema for capacitor energy. As a result, the Companion removes the correct but insufficient transferred knowledge and domain mapping.

The three types of mapping failures demonstrate the limits of DTA's ability to learn domain mappings from the analogy between two worked solutions. As the next section describes, persistent mappings provide one approach to overcoming mapping failures, and we discuss other approaches in Section 6.

4.4.3. Persistent mappings

The results from conditions 3 and 4 of the dynamical analogies experiment demonstrate that persistent mappings support incremental learning of the target domain theory. In condition three, persistent mappings enabled an additional correct retrieval by realigning the content vectors used by MAC/FAC. In addition to aiding retrieval, persistent mappings prevent two of the three types of worked solution mapping failures. Persistent mappings prevent merge failures by requiring the correct correspondence in the worked solution mapping. For example, in the merge failure described in the previous section, the persistent mappings include a required correspondence between `objectTranslating` and `objectRotating` enabling the successful transfer of the definition of kinetic energy by excluding the competing correspondence between `objectThrown` and `objectRotating`. Persistent mappings prevent incomplete mappings by reusing successful aspects of the cross-domain analogy. In the incomplete failure described in the previous section between worked solution Gia-6-28-P and Gia-17-7, the persistent mapping and target domain theory already include the definition of capacitance as a result of the analogy between Phy-1 and Gia-17-24-P. Therefore, when the analogy between Gia-6-28-P and Gia-17-7 transfers the definition of capacitance energy, the Companion is able to use both equations to successfully solve Gia-17-7.

While persistent mappings improved the overall performance of the system, they prevented transfer under certain circumstances. An underlying assumption of persistent mappings is that the entire cross-domain analogy satisfies the one-to-one constraint. That is, each element of the base domain theory corresponds to at most one element in the target domain theory. This assumption was not valid in this experiment. Our representations included more abstraction types for electrical entities than linear mechanical entities. Therefore, when provided with persistent mappings, DTA failed to transfer the domain knowledge necessary to solve the electrical power problem, Gia-18-6, which was successfully solved in the first two conditions. The electrical power equation, $P = Vi$, involves three participants, a power supply, an electrical component, and an electrical conduction event. In the first two conditions, the analogy between the worked solutions for Gia-6-47-P and Gia-18-6 results in a domain mapping including a correspondence between point mass and electrical component. This enables DTA to transfer the power equation, and the Companion to solve the problem. In conditions 3 and 4, the persistent mappings resulting from a successful transfer for problem Gia-21-37-P include a correspondence between `PointMass` and `Inductor-Idealized`. This is a required correspondence during the worked solution analogy between Gia-6-47-P and Gia-18-6, preventing the mapping from including the necessary correspondence between point mass and electrical component. Therefore, an important direction for future work involves relaxing persistent mappings. One approach would be for successful transfers to provide support for individual persistent mappings and failures to provide evidence against them.

5. Related work

The major threads of related work concern computational models of analogy from cognitive science and transfer learning methods from artificial intelligence.

5.1. Cross-domain analogy simulations from cognitive science

Falkenhainer's [5] PHINEAS is the closest system to our own in addressing cross-domain analogical learning. PHINEAS used comparisons of (simulated) behavior to create an initial cross-domain mapping that was subsequently used to create a partial theory for the new domain. It differs, however, in several significant ways: (1) we use analogies between worked solutions to problems to drive the process, (2) we are learning quantitative, rather than qualitative, domain theories, which require very different verification methods, (3) we are using a more psychologically plausible retrieval mechanism, MAC/FAC, and (4) DTA employs persistent mappings to incrementally construct complex cross-domain analogies.

Another analogical problem-solving and learning system is Holyoak and Thagard's PI [25]. PI used a pragmatic theory of analogy to model solving a variation of the radiation problem through schema induction. After a successful analogical problem-solving episode, PI induced a schema which is treated as a general rule that applies to the two analogous situations. PI only used analogy during problem-solving, and its retrieval model was never tested. By contrast, DTA makes analogies between both example problems as well as analogies between domains themselves. The domain theory analogy enables DTA to transfer domain knowledge not explicitly referenced in the particular worked solutions used in creating the domain mapping (e.g., the control knowledge in this experiment). Moreover, DTA verifies its learned knowledge, and uses it to solve new problems from the target domain, whereas PI did neither. Another approach to cross-domain analogy begins by creating visual representations of the source and target [3]. In that work, the authors illustrated how their approach solves the radiation-fortress problem. It is unclear where the visual representation comes from, and how it would perform on non-visual problems such as those presented here.

Another model of analogy, Heuristic-Driven Theory Projection [22], has been used to explore geometric analogical problem-solving [41]. While they have argued that placing this model into an integrated reasoning system is important [42], Heuristic-Driven Theory Projection has not yet been used with other inductive and deductive reasoning methods to date. Scaling up and integration is a major challenge for analogical models [8]. By learning physics domain theories, the transferred knowledge is used in a deductive reasoning task, solving physics problems.

5.2. Transfer learning

Transfer learning consists of acquiring knowledge in one domain or task and using it to improve performance in another domain or task [44]. One example of transfer across tasks involves using SME transfer the effects of actions across different scenarios in a turn-based strategy game [23]. Unlike DTA, in that research analogical mappings only included identical predicates. Persistent mappings were also used in Companions for cross-domain transfer of grid-based strategy games [24]. This work provides evidence for the generality of persistent mappings, given that learning physics domains is quite different than learning strategy games.

Transferring knowledge across different relational domains requires predicate mapping. TAMAR transfers induced rules between Markov Logic Networks through an exhaustive search for predicate mappings between two different relational domains [35]. After an alignment is found, the transferred rules are specialized, relaxed and reweighted to best describe the target domain. While this emphasis on adapting the transferred knowledge during target learning represents an important direction to extend DTA, TAMAR's exhaustive search is psychologically implausible and does not scale to larger relational domains. From a cognitive systems perspective, ICARUS transfer knowledge between tasks through goal decomposition [2] and has been augmented with a representation mapping algorithm [43] to handle cross-domain transfer. The representation mapping methods employed by ICARUS require abstracted domain theories in both the known and new domains. As demonstrated in our experiment, DTA can transfer abstract knowledge from the base domain to the target domain using a domain mapping learned from specific examples.

Within the reinforcement learning [46] community, transfer learning efforts have focused on using mappings between scenarios and domains to transfer state action policies. Lui and Stone [33] use a version of SME to accelerate learning of state action policies in novel but similar tasks within the keep-away soccer domain. Taylor [47] emphasizes the importance of the mapping between the states and actions of the base and target domains. In these systems, the mappings are one shot processes to jump start learning in the target domain. After the initial transfer, the mapping with the base domain is ignored. By using persistent mappings, DTA is an iterative process in which the transferred target knowledge and the domain mapping are incrementally verified and extended. Molineaux et al. [36] present an integrated system, which uses case-based reasoning to select continuous actions within a reinforcement learning framework. Unlike all the above reinforcement learning methods, which use feature vectors, DTA operates over predicate calculus relational descriptions, which facilitates the representation of episodes, explanations and plans. These structures provide additional context for the transfer and enable the transferred knowledge to be more broadly applicable. Furthermore, the physics problem-solving domain provides a clear direction for verification, that is, a transfer failure can be quickly identified by failing to solve the target problem. Our experiments demonstrate how DTA can transfer schemas and control knowledge using only one example from the target domain. For a more direct comparison, it would be necessary to integrate reinforcement learning techniques into DTA.

6. Conclusions and future directions

Using domain general methods for similarity-based retrieval and analogical matching, Domain Transfer via Analogy (DTA) enables the transfer of structured knowledge from an understood domain to a new domain. Persistent mappings support this process by incrementally building up a complex cross-domain analogy from successful mappings.

In this paper, we empirically evaluated DTA across four domains: linear mechanical, rotational mechanical, electrical, and thermal systems. These domains include a wider range of physical phenomena from more dissimilar domains than previous work. Learning such complex domains by analogy does not happen all at once. To enable the incremental accumulation of knowledge about the new domain, we augmented DTA with persistent mappings, demonstrating improved learning performance. Furthermore, the design of this experiment enabled a detailed analysis of the strengths and weaknesses of DTA. The most common cause of failure was the inability to find an analogous example. Once an example has been found, DTA successfully transferred the equation schema necessary to solve the problem in the new domain 78% of the time. Furthermore, we identified three categories of transfer failures: merge failures, one-to-one constraint failures, and incomplete mapping failures. In addition to helping retrieval, we illustrated how persistent mappings address merge failures and incomplete mappings failures.

Our future work with DTA focuses on four directions: integration with domain learning techniques, application to new domains, extending DTA through diagnosis of errors in transferred knowledge, and interactive cross-domain analogy.

6.1. Integration with domain learning techniques

As discussed in the related work section, DTA has not been used with learned domain theories. To perform transfer learning, DTA must be integrated with existing domain learning techniques. For physics problem-solving, a promising direction involves learning generalizations for participant abstraction modeling decisions within each domain [29], then applying DTA to transfer these generalizations to new domains. Because DTA transfers schemas and predicate calculus representations, it should be possible to transfer knowledge produced by a variety of learning algorithms. Previous work has shown the benefits of using reinforcement learning for operator selection within a cognitive architecture [37]. A similar approach could be used to learn a policy for equation selection to augment the control knowledge in this work. DTA should be able to transfer this state action policy to new domains.

6.2. Application to new domains

Another direction for future work is to apply DTA to new domains. Given recent interest in AI and games [31,24], strategy simulations present an interesting domain for exploring the utility of cross-domain analogical learning. These simulations differ in a number of important ways from physics problem-solving (e.g., incomplete knowledge, the integration of planning and action). While cross-domain analogies between strategy games are not as well understood as analogies in physics, it is a reasonable assumption that an agent that performs well in one strategy game or situation would benefit from transferring some aspects of its knowledge to a new game or situation. In addition to game technologies, advances in this area could have applications for training simulations.

6.3. Debugging the transferred knowledge

An underlying assumption of this work is that by identifying similarities between domains, DTA can transfer abstract domain knowledge between them. The existence of these underlying similarities does not necessarily imply that the domains are completely analogous. The cross-domain analogy may provide a useful starting point for the exploration of the new domain, but it may also introduce a number of misconceptions potentially hurting future performance. This would be consistent with psychological findings in student learning [1]. Thus far, persistent mappings have been considered hard constraints for future analogies between the domains. As illustrated in Section 4.4.3, this can have a negative effect on performance. Applying DTA in new domains and over a greater period of learning will likely require relaxing this constraint. This provides an opportunity to exploit either models of conceptual change [14] or model-based diagnosis techniques [4] to identify and repair errors in the persistent mappings or the target domain theory.

6.4. Interactive cross-domain analogy

The fact that cross-domain analogies are used so frequently in textbooks and explanatory discourse implies that these are important communicative devices. DTA represents an important step toward a computational account of the interactive cross-domain analogy process. Suggested correspondences could be treated as additional persistent mappings, informing both the retrieval and mapping process. Furthermore, a user could provide analogous examples allowing DTA to avoid retrieval failures and drastically improves performance as shown by conditions 2 and 4 in our experiment.

A complete account of interactive cross-domain analogy would include the system suggesting insights to the user. This would take the form of suggested correspondences, from which the user could draw new insights, and suggested predictions about aspects of the target domain. For example, in critiquing a course of action in a strategy simulation, the Companion

could express concerns about a particular plan. The ensuing discussion could draw on similarities with other simulations, identifying correspondences between units (e.g., tanks ↔ cavalry) or strategies (e.g., blitzkrieg ↔ scorched earth). Also of importance would be the predictions resulting from the cross-domain analogies. If the player frequently overextends his cavalry in one simulation, and there is a correspondence between cavalry in that simulation and tanks in another simulation, then the system could warn the player to avoid overextending his tanks. In interactive cross-domain analogy, the system informs the user, and the user informs the system. This collaboration extends the knowledge of both participants about the domains and is a worthy target for future AI research.

Acknowledgements

This research was supported by the Cognitive Science Program of the Office of Naval Research. The authors would like the thank David Aha and Danny Bobrow for comments on earlier drafts of this work.

Appendix A

A.1. Gia-4-1 worked solution representation

```
(isa Gia-4-1-WorkedSolution PhysicsWorkedSolution)
(workedSolutionForKBContentTest Gia-Query-4-1 Gia-4-1-WorkedSolution)
(workedSolutionMtForTestMt Gia-4-1 Gia-4-1-WS)
(firstSolutionStep Gia-4-1-WorkedSolution Gia-4-1-WS-Step-1)
(isa Gia-4-1-WS-Step-1 WorkedSolutionStep)
(hasSolutionSteps Gia-4-1-WorkedSolution Gia-4-1-WS-Step-1)
(isa Gia-4-1-WS-Step-1
 CategorizingAPhysicsProblem)
(solutionStepUses Gia-4-1-WS-Step-1
 (isa Gia-4-1 LinearMechanicsProblemCase))
(solutionStepResult Gia-4-1-WS-Step-1
 (isa Gia-4-1 LinearMechanicsProblemCase))
(priorSolutionStep Gia-4-1-WS-Step-2 Gia-4-1-WS-Step-1)
(isa Gia-4-1-WS-Step-2 WorkedSolutionStep)
(hasSolutionSteps Gia-4-1-WorkedSolution Gia-4-1-WS-Step-2)
(isa Gia-4-1-WS-Step-2
 SubstitutingBindingsForVariables)
(solutionStepUses Gia-4-1-WS-Step-2
 (querySentenceOfQuery Gia-Query-4-1
  (valueOf
   (MeasurementAtFn ((QPQuantityFn ForceQuantity) RaceCar-4-1 DefinitionOfNetForce)
    Acc-4-1)?answer)))
(solutionStepUses Gia-4-1-WS-Step-2 (isa RaceCar-4-1 RaceCar))
(solutionStepUses Gia-4-1-WS-Step-2
 (isa Acc-4-1 TransportWithMotorizedLandVehicle))
(solutionStepUses Gia-4-1-WS-Step-2
 (objectTranslating Acc-4-1 RaceCar-4-1))
(solutionStepResult Gia-4-1-WS-Step-2
 (equationForSolution Gia-4-1
  (mathEquals
   (MeasurementAtFn ((QPQuantityFn ForceQuantity) RaceCar-4-1 DefinitionOfNetForce)
    Acc-4-1)
   (TimesFn ((QPQuantityFn Mass) RaceCar-4-1)
    (MeasurementAtFn ((QPQuantityFn Acceleration) RaceCar-4-1)
     Acc-4-1)))))
(solutionStepUses Gia-4-1-WS-Step-2
 (equationFormFor DefinitionOfNetForce
  (mathEquals
   (MeasurementAtFn ((QPQuantityFn ForceQuantity) ?obj DefinitionOfNetForce) ?evt)
   (TimesFn ((QPQuantityFn Mass) ?obj)
    (MeasurementAtFn ((QPQuantityFn Acceleration) ?obj) ?evt)))))
(solutionStepUses Gia-4-1-WS-Step-2
  (abstractionForObject RaceCar-4-1 PointMass))
(solutionStepUses Gia-4-1-WS-Step-2
```

```
  (abstractionForObject Acc-4-1 ConstantTranslationAccelerationEvent))
 (priorSolutionStep Gia-4-1-WS-Step-3 Gia-4-1-WS-Step-2)
 (isa Gia-4-1-WS-Step-3 WorkedSolutionStep)
 (hasSolutionSteps Gia-4-1-WorkedSolution Gia-4-1-WS-Step-3)
 (isa Gia-4-1-WS-Step-3
  SolvingAMathematicalEquation)
 (solutionStepUses Gia-4-1-WS-Step-3
  (equationForSolution Gia-4-1
   (mathEquals
    (MeasurementAtFn ((QPQuantityFn ForceQuantity) RaceCar-4-1 DefinitionOfNetForce)
     Acc-4-1)
    (TimesFn ((QPQuantityFn Mass) RaceCar-4-1)
     (MeasurementAtFn ((QPQuantityFn Acceleration) RaceCar-4-1)
      Acc-4-1)))))
 (solutionStepUses Gia-4-1-WS-Step-3
  (valueOf
   (MeasurementAtFn ((QPQuantityFn Acceleration) RaceCar-4-1) Acc-4-1)
   (MetersPerSecondPerSecond 5)))
 (solutionStepUses Gia-4-1-WS-Step-3
  (valueOf ((QPQuantityFn Mass) RaceCar-4-1) (Kilogram 1500)))
 (solutionStepResult Gia-4-1-WS-Step-3
  (valueOf
   (MeasurementAtFn ((QPQuantityFn ForceQuantity) RaceCar-4-1 DefinitionOfNetForce)
    Acc-4-1)
   (Newton 7500)))
```

A.2. Linear Mechanics Domain Theory Representation

```
(def-encapsulated-history DefinitionOfNetForce
    :participants ((theObject :type PointMass)
                   (theEvent :type ConstantTranslationAccelerationEvent))
    :conditions ((objectTranslating theEvent theObject))
    :consequences
    ((equationFormFor DefinitionOfNetForce
                      (mathEquals
                       (MeasurementAtFn ((QPQuantityFn ForceQuantity)
                                         theObject DefinitionOfNetForce)
                              theEvent)
                       (TimesFn ((QPQuantityFn Mass) theObject)
                              (MeasurementAtFn ((QPQuantityFn Acceleration) theObject)
                                     theEvent))))))

(def-encapsulated-history DefinitionOfTranslationalMomentum
    :participants ((theObject :type PointMass)
                   (theEvent :type ConstantTranslationAccelerationEvent)
                   (theTime :type LMTimeInterval
                           :constraints ((occursDuring theEvent theTime))))
    :conditions ((objectTranslating theEvent theObject))
    :consequences
    ((equationFormFor DefinitionOfTranslationalMomentum
                      (mathEquals (MeasurementAtFn
                                   ((QPQuantityFn LinearMomentum) theObject) theTime)
                                  (TimesFn (MeasurementAtFn
                                            ((QPQuantityFn Speed) theObject) theTime)
                                           ((QPQuantityFn Mass) theObject))))
     (c+ ((QPQuantityFn LinearMomentum) theTime)
         (MeasurementAtFn ((QPQuantityFn LinearMomentum) theObject) theTime)
         DefinitionOfTranslationalMomentum)))

(def-encapsulated-history DefinitionOfTranslationalKineticEnergy
    :participants ((theObject :type PointMass)
```

```
                (theEvent :type ConstantTranslationAccelerationEvent)
                (theTime :type LMTimeInterval
                        :constraints ((occursDuring theEvent theTime))))
    :conditions ((objectTranslating theEvent theObject))
    :consequences
    ((equationFormFor DefinitionOfTranslationalKineticEnergy
                    (mathEquals
                     (MeasurementAtFn
                          ((QPQuantityFn TranslationalKineticEnergy) theObject)
                          theTime)
                     (TimesFn 0.5 ((QPQuantityFn Mass) theObject)
                            (SquaredFn (MeasurementAtFn ((QPQuantityFn Speed) theObject)
                                                        theTime)))))
     (c+ (MeasurementAtFn ((QPQuantityFn TotalEnergy) theObject) theTime)
         (MeasurementAtFn ((QPQuantityFn TranslationalKineticEnergy) theObject)
       theTime)
         DefinitionOfTranslationalKineticEnergy)))

(def-encapsulated-history ConservationOfLinearMomentum
    :participants ((theEvent1 :type ConstantTranslationAccelerationEvent)
                (theEvent2 :type ConstantTranslationAccelerationEvent))
    :conditions ((contiguousAfter theEvent2 theEvent1))
    :consequences ((equationFormFor ConservationOfLinearMomentum
                                (mathEquals ((QPQuantityFn LinearMomentum) theEvent1)
                                        ((QPQuantityFn LinearMomentum) theEvent2)))))

(def-encapsulated-history WorkEnergyTheorem-LM
    :participants ((theObject :type PointMass)
                (theEvent :type ConstantTranslationAccelerationEvent))
    :conditions ((objectTranslating theEvent theObject))
    :consequences
    ((equationFormFor WorkEnergyTheorem-LM
                    (mathEquals ((QPQuantityFn NetWorkQuantity) theObject theEvent)
                             (DifferenceFn
                              (MeasurementAtFn ((QPQuantityFn TotalEnergy) theObject)
                                               (EndFn theEvent))
                              (MeasurementAtFn ((QPQuantityFn TotalEnergy) theObject)
                                               (StartFn theEvent)))))))

(def-encapsulated-history DefinitionOfHockesLaw
    :participants (
                (theSpring :type Spring-Idealized )
                (theDeflection :type Compressing-Linear))
    :conditions ((objectTranslating theDeflection theSpring))
    :consequences
    ((equationFormFor DefinitionOfHockesLaw
                    (mathEquals
                     (MeasurementAtFn
                        ((QPQuantityFn ForceQuantity) theSpring theDeflection)
                        theDeflection)
                     (QuotientFn (MeasurementAtFn
                                   ((QPQuantityFn LinearDeflection) theSpring)
                                   theDeflection)
                              ((QPQuantityFn Compliance-Linear) theSpring)
                              )))))

(def-encapsulated-history DefinitionOfTranslationalEPE
    :participants ((theSpring :type Spring-Idealized)
                (theDeflection :type Compressing-Linear))
    :conditions ((objectTranslating theDeflection theSpring))
    :consequences
```

```
    ((equationFormFor DefinitionOfTranslationalEPE
                      (mathEquals
                       (MeasurementAtFn ((QPQuantityFn TranslationalEPE) theSpring)
                              theDeflection)
                       (TimesFn .5
                              (QuotientFn
                               (SquaredFn
                                (MeasurementAtFn
                                     ((QPQuantityFn LinearDeflection) theSpring)
                                     theDeflection))
                               ((QPQuantityFn Compliance-Linear) theSpring)))))))

(def-encapsulated-history DefinitionOfLinearDamper
    :participants ((theDashPot :type DashPot-Linear)
                (theEvent :type ConstantTranslationAccelerationEvent))
    :conditions ((objectTranslating theEvent theDashPot))
    :consequences ((equationFormFor DefinitionOfLinearDamper
                                (mathEquals
                                 (MeasurementAtFn
                                   ((QPQuantityFn ForceQuantity) theDashPot theEvent)
                                   theEvent)
                                 (TimesFn
                                  ((QPQuantityFn LinearDamperConstant) theDashPot)
                                  (MeasurementAtFn ((QPQuantityFn Speed) theDashPot)
                                         theEvent)
                                         )))))

(def-encapsulated-history DefinitionOfPower-Linear
    :participants ((theObject :type PointMass)
                (theEvent :type ConstantTranslationAccelerationEvent)
                (theForce :type ForceVector
                    :constraints ((applyingAForceOnDuring
                                       theForce theObject theEvent))))
    :conditions ((objectTranslating theEvent theObject))
    :consequences
    ((equationFormFor DefinitionOfPower
                      (mathEquals (MeasurementAtFn
                                   ((QPQuantityFn Power) theObject theForce) theEvent)
                                (TimesFn
                                    (MeasurementAtFn
                                       ((QPQuantityFn ForceQuantity) theObject theForce)
                                      theEvent)
                                    (MeasurementAtFn ((QPQuantityFn Speed) theObject)
                                      theEvent)))
                      )))
```

References

[1] M. Burnstein, Concept formation by incremental analogical reasoning and debugging, in: R.S. Michalski, J.B. Carbonell, T.M. Mitchell (Eds.), Machine Learning: An Artificial Intelligence Approach, vol. 2, Kaufmann, Los Altos, CA, 1986.

[2] D. Choi, T. Konik, N. Nejati, C. Park, P. Langley, Structural transfer of cognitive skills, in: Proceedings of the Eighth International Conference on Cognitive Modeling, Ann Arbor, MI, 2007.

[3] J. Davies, A.K. Goel, Visual analogy in problem solving, in: Proceedings of the International Joint Conference on Artificial Intelligence 2001, Morgan Kaufmann Publishers, 2001, pp. 377–382.

[4] J. de Kleer, J. Kurien, Fundamentals of model-based diagnosis, in: Proceedings of SafeProcess03, 2003.

[5] B. Falkenhainer, Learning from physical analogies, Technical report No. UIUCDCS-R-88-1479, University of Illinois at Urbana-Champaign, 1988 (Ph.D. thesis).

[6] B. Falkenhainer, K. Forbus, D. Gentner, The structure mapping engine: Algorithm and examples, Artificial Intelligence 41 (1989).

[7] M. Fogiel (Ed.), The Physics Problem Solver, Research and Education Association, 1994.

[8] K. Forbus, Exploring analogy in the large, in: D. Gentner, K. Holyoak, B. Kokinov (Eds.), Analogy: Perspectives from Cognitive Science, MIT Press, 2001.

[9] K. Forbus, J. de Kleer, Building Problem Solvers, MIT Press, 1993.

[10] K. Forbus, R. Ferguson, D. Gentner, Incremental structure-mapping, in: Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society, August 1994.

[11] K. Forbus, D. Oblinger, Making SME greedy and pragmatic, in: Proceedings of the Cognitive Science Society, 1990.

[12] K. Forbus, D. Gentner, K. Law, MAC/FAC: A model of similarity-based retrieval, Cognitive Science 19 (2) (1995) 141–205.
[13] K. Forbus, M. Klenk, T. Hinrichs, Companion cognition systems: Design goals and some lessons learned, IEEE-Intelligent Systems 24 (4) (July/August 2009) 36–46.
[14] S. Friedman, K. Forbus, Learning causal models via progressive alignment and qualitative modeling: A simulation, in: Proceedings of the 30th Annual Conference of the Cognitive Science Society (CogSc-08), Washington, DC, 2008.
[15] D. Gentner, Structure-mapping: A theoretical framework for analogy, Cognitive Science 7 (1983) 155–170.
[16] D. Gentner, Why we're so smart, in: D. Gentner, D. Goldin-Meadow (Eds.), Language in Mind: Advances in the Study of Language and Thought, MIT Press, Cambridge, MA, 2003.
[17] D. Gentner, D.R. Gentner, Flowing waters or teeming crowds: Mental models of electricity, in: D. Gentner, A.L. Stevens (Eds.), Mental Models, Greenwich University Press, Eltham, London, 1983.
[18] D. Gentner, M.J. Rattermann, K.D. Forbus, The roles of similarity in transfer: Separating retrievability from inferential soundness, Cognitive Psychology 25 (1993) 524–575.
[19] D. Gentner, S. Brem, R.W. Ferguson, P. Wolff, A.B. Markman, K.D. Forbus, Analogy and creativity in the works of Johannes Kepler, in: T.B. Ward, S.M. Smith, J. Vaid (Eds.), Creative Thought: An Investigation of Conceptual Structures and Processes, American Psychological Association, Washington, DC, 1997.
[20] D. Giancoli, Physics: Principles with Applications, 3rd edition, Prentice Hall, 1991.
[21] M. Gick, K. Holyoak, Schema induction and analogical transfer, Cognitive Psychology 15 (1983) 1–38.
[22] H. Gust, K.-U. Kühnberger, U. Schmid, Metaphors and Heuristic-Driven Theory Projection (HDTP), Theoretical Computer Science 354 (1) (2006) 98–117.
[23] T. Hinrichs, K. Forbus, Analogical learning in a turn-based strategy game, in: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, Hyderabad, India, 2007.
[24] T. Hinrichs, K. Forbus, Transfer learning through analogy in games, AI Magazine 32 (1) (2011) 72–83.
[25] K.J. Holyoak, P. Thagard, A computational model of analogical problem solving, in: S. Vosniadou, A. Ortony (Eds.), Similarity and Analogical Reasoning, Cambridge University Press, New York, 1989.
[26] "Hooke's law, work and elastic potential energy", Physics homework help from experts, 21 March 2009; http://www.physics247.com/physics-homework-help/elastic-potential.php.
[27] M. Klenk, K. Forbus, Measuring the level of transfer learning by an AP physics problem-solver, in: Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI-07), Vancouver, BC, 2007.
[28] M. Klenk, K. Forbus, Domain transfer via cross-domain analogy, in: Special Issue on "Analogies: Integrating Cognitive Abilities", Cognitive Systems Research 10 (3) (2009) 240–250.
[29] M. Klenk, S. Friedman, K. Forbus, Learning modeling abstractions via generalization, in: Proceedings of the 22nd International Workshop on Qualitative Reasoning, Boulder, CO, 2008.
[30] M. Klenk, K. Forbus, E. Tomai, H. Kim, Using analogical model formulation with sketches to solve Bennett mechanical comprehension test problems, in: Special issue on "Test-Based AI", Journal Experimental and Theoretical Artificial Intelligence 23 (3) (2011) 299–327, Taylor & Francis.
[31] J. Laird, M. van Lent, Interactive computer games: Human-level AI's killer application, AI Magazine 22 (2) (2001) 15–25.
[32] P. Langley, The cognitive system paradigm, Advances in Cognitive Systems 1 (2012) 3–13.
[33] X. Lui, P. Stone, Value-function-based transfer for reinforcement learning using structure mapping, in: Proceedings of the 21st Association for the Advancement of Artificial Intelligence (AAAI-06), Boston, MA, 2006.
[34] C. Matuszek, J. Cabral, M. Witbrock, J. DeOliveira, An introduction to the syntax and content of Cyc, in: Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering, Stanford, CA, 2006.
[35] L. Mihalkova, T. Huynh, R.J. Mooney, Mapping and revising Markov Logic Networks for transfer learning, in: Proceedings of 22nd Association for the Advancement of Artificial Intelligence (AAAI) Conference, July 2007, pp. 608–614.
[36] M. Molineaux, D. Aha, P. Moore, Learning continuous action models in a real-time strategy environment, in: Proceedings of FLAIRS-08 Conference, 2008.
[37] S. Nason, J.E. Laird, Soar-RL: Integrating reinforcement learning with soar, Cognitive Systems Research 6 (1) (2005) 51–59.
[38] K. Ogata, System Dynamics, Prentice Hall, 1997.
[39] H. Olson, Dynamical Analogies, D. Van Nostrand, 1943.
[40] Y. Pisan, An integrated architecture for engineering problem solving, Science, PhD thesis, Department of Computer, Northwestern University, 1998.
[41] A. Schwering, C. Bauer, I. Đorčeva, H. Gust, U. Krumnack, K.-U. Kühnberger, The impact of gestalt principles on solving geometric proportional analogies, in: 2nd International Analogy Conference (ANALOGY09), Sofia, Bulgaria, 2009.
[42] A. Schwering, U. Krumnack, K.-U. Kühnberger, H. Gust, Analogy as integrating framework for human-level reasoning, in: The Proceedings of the 1st Conference on Artificial General Intelligence (AGI-08), IOS Press, Memphis, TN, 2008, pp. 419–423.
[43] D. Shapiro, T. Könik, P. O'Rorke, Achieving far transfer in an integrated cognitive architecture, in: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, Chicago, IL, 2008.
[44] D. Shapiro, D.J. Stracuzzi, H. Munoz-Avila (Eds.), Special Issue on Transfer of Structural Knowledge, AI Magazine 32 (1) (2011).
[45] J. Shearer, A. Murphy, H. Richardson, Introduction to System Dynamics, Addison–Wesley, Reading, MA, 1971.
[46] R. Sutton, A. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, 1998.
[47] M. Taylor, Transfer in Reinforcement Learning Domains, Studies in Computational Intelligence, vol. 216, Springer-Verlag, Berlin, 2009.